IMPUTING INCOMPLETE SOFTWARE METRIC DATABASE USING ITERATIVE NEAREST NEIGHBOUR BASED ALGORITHM

Ito Wasito[1] and Nurul Farihan Ab Azid

[1]Department of Software Engineering
Faculty of Computer Science and Information Systems
University Teknologi Malaysia
81300 Skudai, Johor

[2]Department of Electrical and Computer Engineering
Faculty of Engineering
International Islamic University Malaysia
Jl. Gombak, Kuala Lumpur

Email: [1]ito@utm.my

Abstract. Data sets are playing an important role not only for the construction of effort precision model but also in building software cost estimation models. Knowing the estimated cost at the early project would constitute so much to the software development cycle. Therefore, with the existence of missing values in the data sets, it will be thorny to achieve all these. The purpose of this paper is to review and implement the iterative nearest neighbour based data imputation technique for handling incomplete software metric database. The experiments are carried out using real database for the input of the algorithm for the purpose of relating the missingness data in the real database to examine the accuracy of the imputed data compared to the actual data.

Keywords: Software Metric, Nearest Neighbour Imputation, Software cost estimation, Missing Values, Least Squares.

1. INTRODUCTION

In software engineering databases, the problem of missing data often encountered and it will become harder to construct effort precision model with the existence of missing data. The International Software Benchmarking Standards Group (ISBSG) database also having the same problem whereby more than 40 percent of their variables gone missing [4,7].

Data sets are playing an important role not only for the construction of effort precision model but also in building software cost estimation models [5]. This is specifically

for software engineering people. Knowing the estimated cost at the early project would constitute so much to the software development cycle. "Accurate cost estimates would allow organizations to make more realistic bids on external contracts" [6]. Therefore, with the existence of missing values in the data sets, it will be thorny to achieve all these.

There are several factors that lead to the missing values in the data sets. "High data collection cost may cause missing data" [4]. This is due to the cost of gathering and reporting the data. Besides, there are some data that are costly and difficult to collect. For instance, in Enterprise Resource Planning (ERP) project, the data on Interfaces and Effort hard to collect compared to the data on Users, Sites and Modules.

Another relevant factor is the presence of wild values. A wild value is a value that we know untrue or illogical to be there. Possibly that is caused by punching error or it is being reported wrongly by the person who do not know how to measure (human error). The solution for this matter simply leaves the observations empty and thus would create another 'missing' values. Other reason is that some respondents did not respond to all questions in questionnaire accordingly. The causes behind that are either, they running out of time, lack of understanding of questions, they do not possess adequate knowledge in some of the questions or they simply do not wish to answer those questions which they think might reveal certain information to others. Hence, some of the variables went missing as well. The impact of missing values in the data sets would direct to the loss of information and biasness in the data analysis and hence produce inaccurate model.

2. ITERATIVE MAJORIZATION LEAST-SQUARES (IMLS) ALGORITHM

This method is generated from the weighted least-squares minimization problem can be addressed as a series of non-weighted least squares minimization problems with iteratively adjusting found solutions according to a so-called majorization function.

The following algorithm has been developed by Kiers [3]. This algorithm starts with a complete data matrix and updates it by relying on both non-missing entries and estimates of missing entries.

This algorithm operates with a completed version of matrix $\mathbf{X}$ denoted by $\mathbf{X}^S$ where $s = 0,1,\ldots$ is the iteration's number. At each iteration s, the algorithm finds one best factor of the SVD decomposition of $\mathbf{X}^S$ and imputes the results into the missing entries after which the next iteration starts [1,9,10].

Kiers Algorithm

1. Set $\mathbf{c}' = (1,...,1)$ and normalize it.
2. Set $s=0$ and define matrix $\mathbf{X}^S$ by putting zeros into missing entries of $\mathbf{X}$. Set

a measure of quality $h_i = \sum_{i=1}^{N}\sum_{k=1}^{n} x_{ik}^{s2}$

3. Find the first singular triple $\mathbf{z}_1$, $\mathbf{c}_1$, μ for matrix $\mathbf{X}^S$ by applying the iterative SVD algorithm with $p=1$ and denote the resulting value of criterion (2.4.6) by h_{s+1}. (Vectors $\mathbf{z}_1$, $\mathbf{c}_1$ are assumed normalized here.)
4. If $|h_s - h_{s+1}| > \epsilon$ for a small $\epsilon > 0$, set $s = s+1$, $\mu z_{i1}\, c_{1k}$ for any missing entry (i,k) in $\mathbf{X}$ and go back to step 3.
5. Set $\mu\mathbf{z}_1$ and $\mathbf{c}_1$ as the output.

Now IMLS can be formulated with its properties yet to be explored [2,9,10].

IMLS algorithm

0. Set the number of factors p.
1. Set iteration number $t = 1$.
2. Apply Kiers algorithm to matrix $\mathbf{X}$ with the missing structure $\mathbf{M}$. Denote results by $\mathbf{z}_t$ and $\mathbf{c}_t$.
3. If $t = p$, go to step 5.
4. For (i,k) such that $m_{ik} = 1$, update $x_{ik} = x_{ik} \_ c_{tk}z_{it}$, put $t = t + 1$ and go to step 2.
5. Impute the missing values x_{ik} at $m_{ik} = 0$ with $e_{ik} = 0$.

3. INI: NEAREST NEIGHBORS APPROACH BASED ON IMLS NEAREST NEIGHBORS ALGORITHM

INI algorithm introduced by Wasito and Mirkin [9,10]. In this approach, the imputations are carried out sequentially, by analyzing entities with missing entries one-by-one. An entity containing one or more missing entries, which are to be imputed, is referred to as target entity. According to this approach, a distance measure is computed between the target entity and each of the other and then K entities nearest to the target are selected. See [9,10] for details of the algorithm.

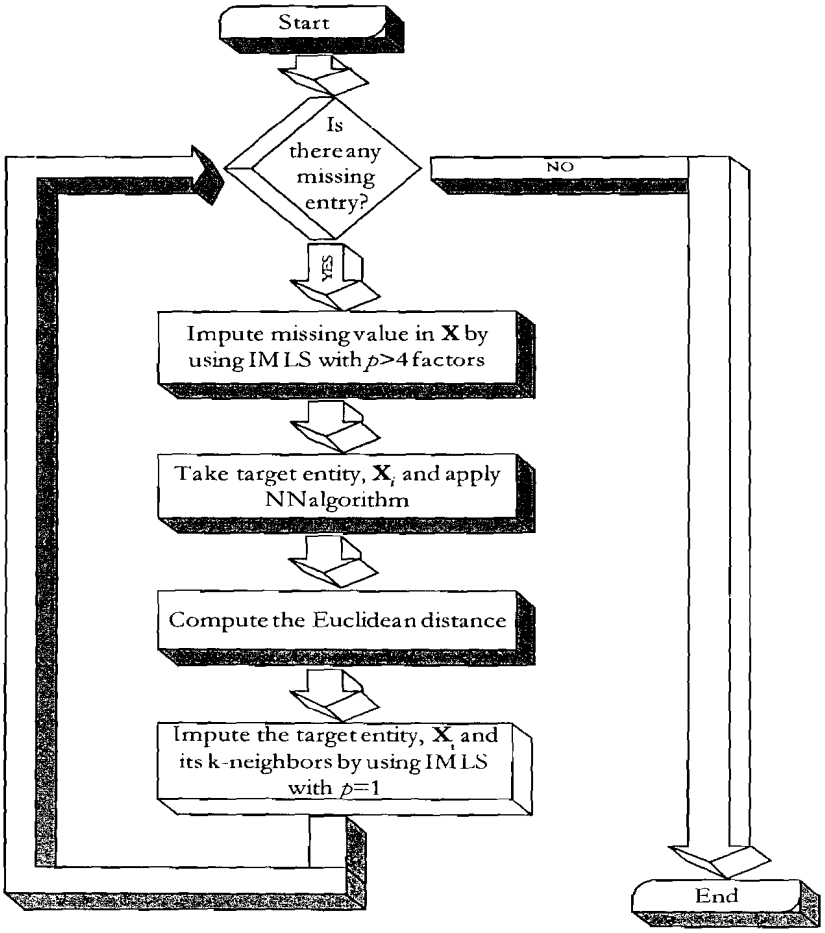

Fig. 2 Flowchart of INI Algorithm

4. EXPERIMENTS

4.1 Data Sets.

Real database is taken from the factual data related to the engineering field and for our project, the database known as the summary of OO Project Metrics for C++. While simulation input we use *rand()* function to randomly generate the input between the specified range.

4.2 Missingness Structure

It is an m-by-n matrix that consists of 0's and 1's which is denoted as *mis*. From the missingness structure, we can have an overall picture of the missing entries that we have in our data set.

In the real database, we have specified the missing entry by '-9'. This '-9' will be replaced by '0' in the missingness structure. Other than that, it will return as '1'. However in the simulation input, the missingness structure is generated randomly by *randsrc()* function in the Matlab. By using this function, we specified the 0's to be merely about 1%, the rest would be 1's. In other words, the total number of '1' is 99% while the total number of '0' is only 1% from the total number of data.

The missingness structure is always updated and each time one missing entry has been imputed, the level of its position is no longer '0' but rather it will be replaced by '1'. From it we are able to know when we should stop imputing the missing entries. This is when we get the missingness structure consists all 1's.

4.3 RESULTS AND DISCUSSIONS

The follows example of actual database that we used to test the output for INI algorithm and we named this input file as 'method.txt'. However, some alterations were made in order to test the accuracy of the imputation values by replacing a few of the data entries with '-9' (indicates missing). After that, we observed the respected outputs and compare with the actual values.

Table 1 Actual - RFC < 5 * #methods (Percentages)

Project ID	<= 5	> 5	<= 5	> 5
C100a	134	6	95.7%	4.3%
C100b	193	8	96.0%	4.0%
C100c	344	11	96.9%	3.1%
C100d	548	14	97.5%	2.5%
C101	1,814	152	92.3%	7.7%
C102a	4,936	171	96.7%	3.3%
C102b	4,865	169	96.6%	3.4%
C102c	4,124	147	90.3%	3.2%

The new-look of input file appears to be as follows:

Table 2 Modified - RFC < 5 * #methods (Percentages)

Project ID	<= 5	> 5	<= 5	> 5
C100a	-9	6	95.7%	4.3%
C100b	193	-9	96.0%	4.0%
C100c	344	11	96.9%	3.1%
C100d	548	14	97.5%	2.5%
C101	1,814	152	92.3%	7.7%
C102a	4,936	171	-9	3.3%
C102b	4,865	169	96.6%	3.4%
C102c	4,124	147	90.3%	-9

Table 3 Project Metrics

Project ID	# Files	# Classes	# NTLC	LOC	Pre Process	Blank	Comm.	CP	NCNB	Exec Stmts
C100a	322	140	81	57086	3620	12250	17664	39.4%	29014	14037
C100b	481	201	124	92231	6449	20019	28058	38.9%	47770	23002
C100c	735	355	204	167541	11264	34612	49940	37.6%	89743	42756
C100d	1193	562	297	261260	20932	50862	78104	37.1%	145648	64935
C101	3227	1966	611	838128	43552	108335	290628	39.8%	443824	175736
C102a	6735	5107	2297	2062982	80193	469213	523035	32.8%	1102727	506790
C102b	7292	5035	2294	2129555	89532	483859	542214	32.9%	1137418	513001
C102c	5975	4566	2095	1948354	79184	434469	485953	32.1%	1058117	481954

Another database called as 'project.txt' was also examined as shown in the table below.

In addition to 'method.txt' and 'project.txt' input files, we also have tested the actual database with the real missing values. This database named as 'meth.txt'. At this instant, we notice how important the imputation methods would be. If at this stage, no efforts taken to overcome this problem, no analysis on the data can be made.

Table 4 Methods

Project ID	0-5	6-15	16-25	26-35	36-45	46-55	56-65	66-75	> 75
C100a	29	67	31	4	4	5	-	-	-
C100b	56	94	36	6	4	5	-	-	-
C100c	80	183	65	15	6	4	2	-	-
C100d	124	293	101	29	8	6	1	-	-
C101	644	628	477	29	27	49	51	29	32
C102a	2,341	1,264	943	282	139	50	33	16	36
C102b	2,325	1,200	946	287	139	52	28	19	38
C102c	1,970	1,032	808	256	114	33	19	13	23

Output:

Below is the output when only the entity in cell [1][2] was replaced by '-9' and the output was approximated to be 5.3022 while the actual value is 6. The output is indicated in the Fig. 3.

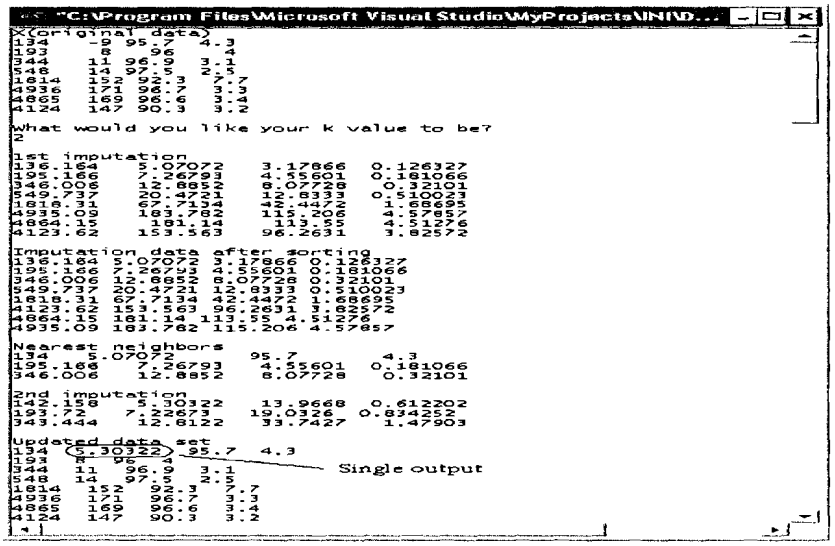

Fig. 3 Output Window: Single Missing Value

Below are the outputs when multiple missing values are found in the original data set which is shown in Table 2. The respected outputs for the particular missing entries are shown in the Fig. 4.

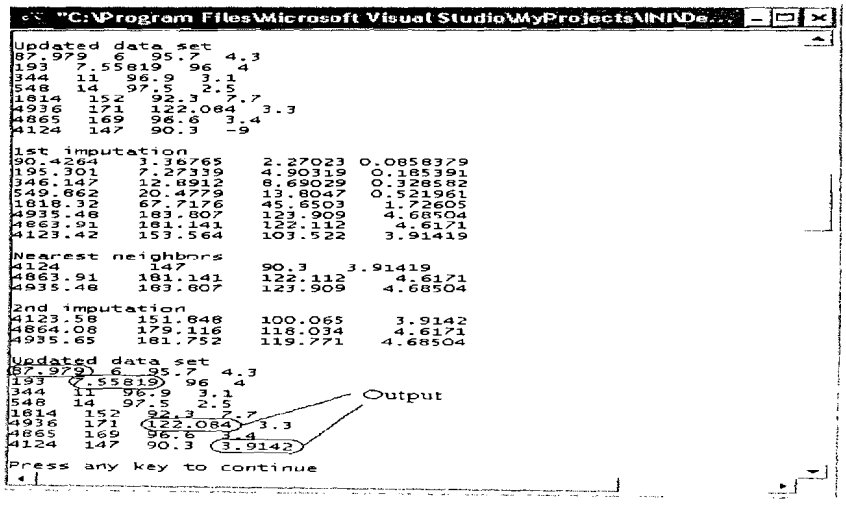

Fig. 4 Output Window: Multiple Missing Values

Fig. 5 shows the output of imputation values for the real missing entries found in the real database. Therefore the missingness possess in the date set will be filled up with designated values shown below.

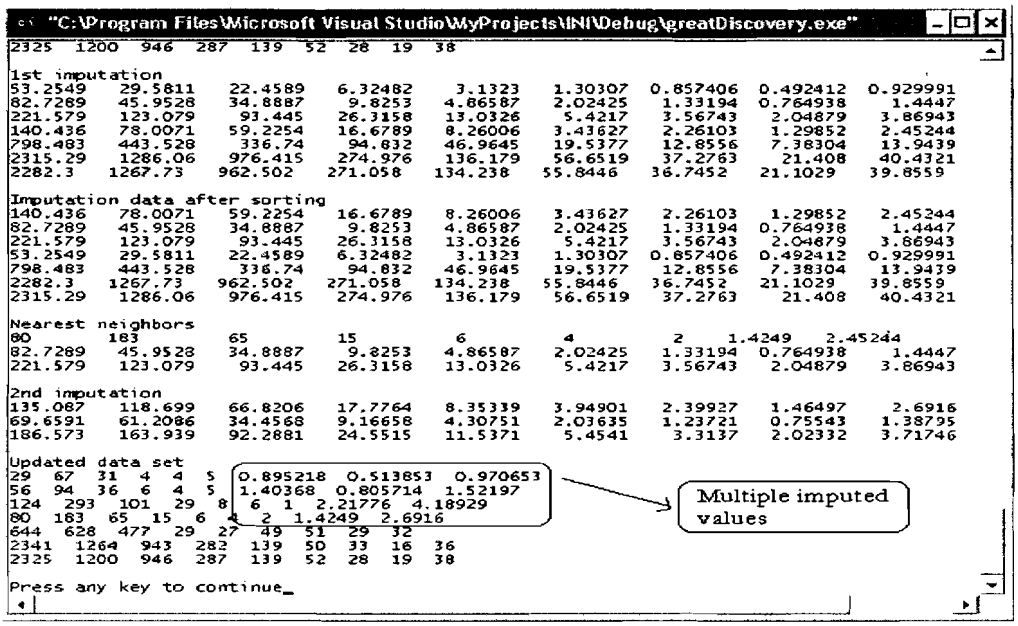

Fig. 5 Output Window: Real Missing Entries

In finding the input-output relationship, we can obtain the percentage difference between the actual value and its imputed one. The equation used throughout the calculation is

$$\% \text{ Difference} = (X_{\text{imputed}} - X_{\text{actual}})^2 \; / \; (X_{\text{actual}})^2 \times 100\%.$$

The percentage difference that can be tolerated is less than 10%. It means that if the difference shows the percentage less than 10%, the imputed value had give the best approximation of the actual value.

Table 5 Input-Output Relationship of the Database in Table 3

Project ID		Actual Data	Imputed Data	% Difference
C100a	**# Files**	322	188.359	17.23%
	LOC	57086	44499.6	**4.86%**
	Comm.	17664	14567.4	**3.07%**
	NCNB	29014	30854.1	**0.40%**
C100c	**# Classes**	355	403.657	**1.88%**
	Pre Process	11264	6977.86	14.48%
	CP	37.6	3.14532	83.97%
	Exec. Stmts.	42756	40878.4	**0.19%**
C101	**# Files**	3227	2727.31	**2.40%**
	# NTLC	611	907.778	23.59%
	Blank	108335	190968	58.18%
	NCNB	443824	447882	**0.008%**
C102b	**# Classes**	5035	4738.87	**0.35%**
	LOC	2129555	2088610	**0.037%**
	Comm.	542214	550475	**0.023%**
	Exec. Stmts.	513001	516308	**0.004%**

Table 5 has shown the comparison between the actual value and its imputed value of some of the data entries in the database found in Table 3. Although some of other entries are not

included in the calculation but from the overall picture of the above table, we can observed that most of the percentage differences are below than 10% (percentage tolerance), even less than 5% and that are make them even look better. Out of 16 entries, 11 entries give a very good approximation compared to the actual value. In Table 5, they are marked up in bold. In opposite, only 5 entries indicate the values that are not in favor with the actual values.

5. CONCLUSIONS AND FUTURE WORKS

We successfully implement INI algorithm which comprised of IMLS algorithm and k-Nearest Neighbors algorithm. Apart from that, the analysis was made based upon the results of two different kinds of inputs which are real database. From the findings, we found that this method had given mostly a good approximation of the missing entries. However, there are certain cases where the imputed values gave a poor estimation. For example, in the case of database that has very large variation between the entries within the same column. With regard to the issues raised, directions for future work should include the following:

(1) A theoretical investigation should be carried out on the properties of convergence for major iterative technique IMLS.

(2) The performances of the least-squares-based techniques should be compared with those of another set of popular imputation techniques based on the maximum likelihood principle. This requires rearranging the setting of experiments since the latter methods are computationally expensive and may fail to converge. In our preliminary experiments, though, the errors were similar between the global least-squares imputation techniques and standard expectation–maximization techniques implementing the maximum likelihood principle.

REFERENCES

[1] Golub, G. H., Charles F.Van Loan. "*Matrix Computation*"*s*. 3rd Edition. The Johns Hopkin University Press. 1996.

[2] Heiser,W.J., "Convergent computation by iterative majorization: theory and applications in multidimensional analysis". In: Krzanowski,W.J. (Ed.), Recent Advances in Descriptive Multivariate Analysis. Oxford University Press, Oxford, pp. 157–189, 1995.

[3] Kiers, H.A.L. "Weighted least squares fitting using ordinary least squares Algorithms", Psychometrika 62 , 251–266, 1997.

[4] Myrtveit, I. , Stensrud, E. and Olsson, I. "Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood Based Method"*s*. IEEE Tanssaction on Softtware Engineering, 27, 11, 2001.

[5] Olinsky, A., Shaw Chen & Harlow, L. "The comparative efficacy of imputation methods for missing data in structural equation modeling". European Journal of Operational Research 151, 1, 2003.

[6] Strike, K. El Emam, K. and Madhavji, N. Software Cost Estimation with Incomplete Data"". IEEE Transacation on Software Engineering 27, 10, 2001.

[7] Cartwright, MH, Shepperd, M.J. and Song, Q. ""Dealing with Missing Software " Project Data. Software Metrics Symposium, Proceedings. Ninth International IEEE, 2003.

[8]. Gass, S.I, T. Rapcs_ak. "Singular value decomposition in AHP. European Journal in Operational Research 154, 2002.

[9]. Wasito, I and Mirkin, B. "Nearest Neighbour approach in the least-squares data imputation algorithms". Information Sciences 169, 1-25, 2005.

[10] Wasito, I and Mirkin, B. "Nearest Neighbour in the least-squares data imputation algorithms with different missing pattern". Computational Statistics and Data Analysis 50, 926-949, 2006.